22883

1   This application is submitted in the name of the following inventors:

2

3	Inventor	Citizenship	Residence City and State
4	Hitz, David	United States	Portola Valley, California
5	Borr, Andrea	United States	Los Gatos, California
6	Hawley, Robert	United States	San Jose, California
7	Muhlestein, Mark	United States	Morgan Hill, California
8	Pearson, Joan	United States	Menlo Park, California

9

10   The assignee is Network Appliance, Inc., a corporation having an office at

11   2770 San Tomas Expressway, Santa Clara, CA 95051.

12

13   Title of the Invention

14

15   File Access Control in a Multi-protocol File Server

16

17   Background of the Invention

18

19   *1.*   *Field of the Invention*

20

21   The invention relates to file access control in a multi-protocol file server.

22

1   *2.   Related Art*

2

3   In an integrated computer network, it is desirable for multiple client devices

4   to share access to the same files. One known method is to provide a network file server

5   for storing files, capable of receiving and responding to file server requests from those

6   client devices. These file server requests are made using a file server protocol, which is

7   recognized and adhered to by both the file server and the client device. Because the files

8   are stored at the file server, multiple client devices have the opportunity to share access to

9   the same files.

10

11   In a file system intended for use by more than one user, it is desirable to re-

12   strict access by programs to files in the file system. Restricting access includes at least

13   the aspects of (1) user authentication—determining that requesting users are truly who

14   they say they are, and (2) access control validation—determining that an authenticated

15   user is allowed to access a particular file in a particular way. When the file system is

16   maintained on a file server remote from the user making the request, there is an additional

17   aspect of the access control protocol—what requests can be made by the user to access

18   files or to set access control for files.

19

20   One problem in the known art is that there are multiple diverse models for

21   access control validation, each typically associated with a particular file system, and there

22   are multiple diverse access control protocols, each typically corresponding to a model for

1 access control validation. Despite the differences between these models and protocols,

2 the file server should respond to file server requests from each user, and should exhibit

3 access control validation behavior, consistent with each user's model and without secu-

4 rity violations or surprises for users.

5

6   For example, a first access control model in common use is associated with

7 the Unix operating system (or a variant thereof). This first access control model associ-

8 ates permissions with each file for a file owner, an owner's group, and all other users.

9 These permissions allow access (for the owner, group, or all other users) to read, write, or

10 execute the indicated file. This first access control model is typically implemented by the

11 NFS ("Network File System") file server protocol, possibly augmented with an adjunct

12 file-locking protocol, NLM ("Network Lock Manager"). A second access control model

13 in common use is associated with the Windows NT operating system. This second access

14 control model associates an ACL (access control list) with each file, each entry in the

15 ACL specifying an individual user, a group of users, or all users. Each entry can allow

16 access (for the specified users) to read, write, or execute the indicated file, or can specifi-

17 cally deny access. This second access control model is typically implemented by the

18 CIFS ("Common Internet File System") protocol. However, NT devices can also use the

19 NFS protocol by means of the "PC NFS" implementation, and Unix devices can also ma-

20 nipulate POSIX ACLs. These two access control models in common use differ in sig-

21 nificant ways, including (1) what permissions can be assigned to a file, (2) with what

1   granularity of specificity permissions can be assigned, and (3) how users are identified so

2   as to match them with permissions.

3

4   One method known in the art is to provide a multi-protocol file server that

5   maps all security semantics to that of a single native operating system for the file server,

6   and uses that single native operating system to validate file access control. The "Samba"

7   system and similar emulation packages are believed to use this known method. This

8   known method has the drawback that it can result in security errors or surprises for those

9   client devices using security semantics other than the file server's native operating sys-

10   tem.

11

12   Another method known in the art is to provide a multi-protocol file server

13   that supports differing types of security semantics for differing files, but attempts to vali-

14   date file access control for each user using the user's access control model. Some "Net-

15   ware" products available from Novell Corporation are believed to use this known

16   method. This known method has the drawback that the user's access control model can

17   differ significantly from the access control model set for the file, resulting in security er-

18   rors or surprises for those client devices using security semantics other than associated

19   with the target file.

20

21   Accordingly, it would be desirable to provide a method and system for en-

22   forcing file security semantics among client devices using multiple diverse access control

1 models and multiple diverse file server protocols. This advantage is achieved in an em-

2 bodiment of the invention in which a multi-protocol file server identifies each file with

3 one particular access control model out of a plurality of possible access control models,

4 and enforces that particular access control model for all accesses to that file. When the

5 file server receives a file server request for that file using a file server protocol with a dif-

6 ferent access control model, the file server translates the access control limits imposed by

7 the file's access control model into no-less-restrictive access control limits in the different

8 access control model. The file server restricts access to the file using the translated ac-

9 cess control limits.

10

11 Summary of the Invention

12

13 The invention provides a method and system for enforcing file access con-

14 trol among client devices using multiple diverse access control models and multiple di-

15 verse file server protocols. A multi-protocol file server identifies each file with one par-

16 ticular access control model out of a plurality of possible models, and enforces that one

17 particular model for all accesses to that file. When the file server receives a file server

18 request for that file using a different access control model, the file server translates the

19 access control limits for that file into no-less-restrictive limits in the different model. The

20 file server restricts access by the client device using the translated access control limits.

21

1   In a preferred embodiment, each file is assigned the access control model of

2   the user who created the file or who last set access control limits for the file. When a user

3   having a different access control model sets access control limits, the access control

4   model for the file is changed to the new model. Files are organized in a tree hierarchy, in

5   which each tree is limited to one or more access control models (which can limit the abil-

6   ity of users to set access control limits for files in that tree). Each tree can be limited to

7   NT-model-only format, Unix-model-only format, or mixed NT-or-Unix-models format.

8

9

Brief Description of the Drawings

10

11   The figure shows a block diagram of a system for enforcing diverse access

12   control models among client devices.

13

14   Detailed Description of the Preferred Embodiment

15

16   In the following description, a preferred embodiment of the invention is de-

17   scribed with regard to preferred process steps and data structures. However, those skilled

18   in the art would recognize, after perusal of this application, that embodiments of the in-

19   vention may be implemented using one or more general purpose processors (or special

20   purpose processors adapted to the particular process steps and data structures) operating

21   under program control, and that implementation of the preferred process steps and data

1　structures described herein using such equipment would not require undue experimenta-

2　tion or further invention.

3

4　　　　　Inventions described herein can be used in conjunction with inventions de-

5　scribed in the following applications:

6

7　o　Application Serial No. 08/985,398, filed December 5, 1997, in the name of Andrea

8　　　Borr, titled "Multi-Protocol Unified File-locking", attorney docket number NET-

9　　　023.

10

11　　　This application is hereby incorporated by reference as if fully set forth

12　herein.

13

14　　　The enclosed technical appendix is part of the disclosure of the invention,

15　and is hereby incorporated by reference as if fully set forth herein.

16

17　*System Elements*

18

19　　　The figure shows a block diagram of a system for enforcing diverse access

20　control models among client devices.

21

22　　　A system 100 includes a file server 110, and a set of client devices 120.

1

2   The file server 110 maintains a file system 111, including a set of files 112.

3

4   In a preferred embodiment, the file server 110 maintains the file system 111

5 using inventions described in the following patent applications:

6

7 o   Application Serial No. 08/471,218, filed June 5, 1995, in the name of inventors

8   David Hitz et al., titled "A Method for Providing Parity in a Raid Sub-System

9   Using Non-Volatile Memory", attorney docket number NET-004;

10

11 o   Application Serial No. 08/454,921, filed May 31, 1995, in the name of inventors

12   David Hitz et al., titled "Write Anywhere File-System Layout", attorney docket

13   number NET-005;

14

15 o   Application Serial No. 08/464,591, filed May 31, 1995, in the name of inventors

16   David Hitz et al., titled "Method for Allocating Files in a File System Integrated

17   with a Raid Disk Sub-System", attorney docket number NET-006.

18

19   Each of these applications is hereby incorporated by reference as if fully set

20 forth herein.

21

1   The file server 110 is disposed for receiving file server requests 121 from

2   the client devices 120. The file server 110 parses each request 121, determines whether

3   the operation requested in the request 121 is allowed (for the client device 120 that sent

4   the request 121 and for the one or more target files 112 specified by the request 121). If

5   allowed, the file server 110 performs that operation on the one or more target files 112.

6

7   The file server 110 is also disposed for transmitting file server responses

8   122 to the client devices 120. The file server 110 determines the response to each request

9   121 (possibly including a response indicating that the requested operation was not al-

10 lowed), generates that response 122, and transmits that response 122 to the client device

11 120 that sent the request 121.

12

13   Each client device 120 is disposed for transmitting file server requests 121

14 to the file server 110, and for receiving file server responses 122 from the file server 110.

15

16 *Access Control Models*

17

18   In a preferred embodiment, each client device 120 can be either a Unix cli-

19 ent device 120 or a Windows NT client device 120. Each client device 120 can either use

20 the NFS file server protocol to make requests 121, or use the CIFS file server protocol to

21 make requests 121. (Although typically Unix client devices 120 use the NFS file server

22 protocol and NT client devices 120 use the CIFS file server protocol, it is possible for NT

1   client devices 120 to use the NFS file server protocol by using the PC-NFS implementa-

2   tion of that file server protocol.) The file server 110 receives each request 121 and (if

3   allowed) performs the requested operation on the target files 112 specified by the request

4   121.

5

6   The file server 110 supports more than one access control model, including

7   a "Unix Perms" access control model (herein "Unix security style") and an "NT ACL"

8   access control model (herein "NT security style").

9

10   Unix security style uses user IDs (UIDs) to identify users, and group IDs

11   (GIDs) to identify groups to which those users belong. Unix security style associates the

12   following access control limits with each file:

13

14   o   a UID for the owner;

15

16   o   a GID for the owner;

17

18   o   a set of "user" permissions—allowing permission to read, write, or execute the file

19   by the owning user;

20

21   o   a set of "group" permissions—allowing permission to read, write, or execute the

22   file by the owning user's group; and

1

2   o   a set of "other" permissions—allowing permission to read, write, or execute the

3   file by all other users.

4

5   Unix security style is supported by the NFS ("Network File System") file

6   server protocol, possibly augmented with the NLM ("Network Lock Manager") adjunct

7   file-locking protocol.

8

9   NFS is a stateless protocol, so each NFS file server request 121 includes the

10   UIDs and GIDs of the user making the request. The Unix client device 120 determines

11   the UIDs and GIDs for the user at a login time for the user, by reference to the system

12   password file (/etc/passwd) and group file (/etc/groups).

13

14   To enforce file access control using Unix security style, the file server 110

15   determines if the request 121 is from the owning user, from a user in the owning user's

16   group, or from some other user. Responsive to this determination, the file server 110

17   uses one of the user permissions, the group permissions, or the other permissions, to de-

18   termine whether to allow the request 121.

19

20   NT security style uses security IDs (SIDs) to identify both users and

21   groups. NT security style associates the following access control limits with each file:

22

1   o   an SID for the owner;

2

3   o   an SID for the owner's group;

4

5   o   an ACL (access control list).

6

7   The NT ACL includes one or more ACEs (access control entries), each of

8   which includes an SID indicating the user or group to which it applies, and a set of per-

9   missions. NT security style provides for the three Unix permissions (read, write, or exe-

10   cute), as well as "CHANGE PERMISSIONS" permission, "TAKE OWNERSHIP" per-

11   mission, "DELETE" permission, "DELETE CHILD" permission, and other permissions.

12

13   NT security style is supported by the CIFS ("Common Internet File Sys-

14   tem") protocol. NT security style is further described in the following articles: R. Rei-

15   chel, "Inside Windows NT Security", Windows/DOS Developers' Journal (April & May

16   1993), and in Stephen Sutton, "Windows NT Security Guide" (ISBN 0201419696).

17

18   CIFS is a session-based protocol, so the NT client device 120 transmits the

19   NT user name and password to the file server 110 at a session connect time, from which

20   the SIDs for the user and the user's groups are determined. The file server 110 can at-

21   tempt to authenticate the user itself, or (preferably) forward the NT user name and pass-

22   word to an NT primary domain controller.

1

2   To enforce file access control using NT security style, the file server 110

3   determines the SID for the user and the user's group, for the user making the request 121.

4   The file server 110 accumulates permissions granted to that user from ACEs that apply,

5   then subtracts permissions specifically denied to that user. Responsive to this accumula-

6   tion and subtraction, the file server 110 determines whether to allow the request 121.

7

8   Although a preferred embodiment of the invention is described with regard

9   to Unix security style and NT security style, the invention can readily be used with other

10   access control models, such as the "POSIX ACL" access control model supported by

11   some Unix devices, and by some other operating systems. The concepts and features of

12   the invention described herein can readily be used in a file server 110 supporting the

13   "POSIX ACL" access control model in addition to or instead of the access control models

14   described in detail herein, without further invention or undue experimentation. Accord-

15   ingly, the scope and spirit of the invention includes such file servers and methods for

16   their use.

17

18   The file server 110 designates each file 112 maintained in its file system

19   111 as having one specific access control model out of the plurality of access control

20   models it supports. In a preferred embodiment, each file 112 is designated to use either

21   Unix security style or NT security style. The file server 110 enforces the designated se-

22   curity style for each file 112 for all attempts to access that file 112. Thus, the file server

1   110 enforces the designated security style for all requests 121 made for that target file

2   112, whether those requests 121 come from Unix devices or NT devices, and whether

3   those requests 121 use the NFS file server protocol or the CIFS file server protocol.

4

5   *Access Control Enforcement*

6

7   If the file server 110 receives a request 121 for a target file 112, and the re-

8   quest 121 matches the security style target file 112, the file server 110 validates the re-

9   quest 121 against access control limits for that file 112 imposed by that security style.

10

11   The file server 110 thus recognizes and enforces at least the following cir-

12   cumstances:

13

14   o   NT security style. The file 112 has NT security style and has a corresponding set

15   of access control limits (an NT ACL), which has been set by a client device 120

16   using the CIFS file server protocol.

17

18   If a client device 120 makes a request 121 to access the file 112 using the

19   CIFS file server protocol, the file server 110, if it is able to, enforces the NT ACL using

20   NT security style.

21

14

1   If the file server 110 is able to determine the NT user, either by communi-

2   cation with an NT domain controller, or by reference to an NT user SID (security ID)

3   database, the file server 110 is able to enforce the NT ACL using NT security style.

4

5   If the file server 110 is not able to determine the NT user, it determines the

6   equivalent Unix user, using a UID for a Unix user recorded for the CIFS file server pro-

7   tocol session, and enforces the NT ACL as if the request 121 came from that Unix user.

8

9   o   Unix security style. The file 112 has Unix security style and has a corresponding

10   set of access control limits (Unix Perms), which have been set by a client device

11   120 using the NFS file server protocol

12

13   If a client device 120 makes a request 121 to access the file 112 using the

14   NFS file server protocol, the file server 110 enforces the Unix Perms using Unix security

15   style.

16

17   However, the file server 110 can also receive a request 121 that does not

18   match the security style for the target file 112. The file server 110 can enforce the secu-

19   rity style for the target file 112 against a non-matching client device 120 by validating

20   either (1) a translated user ID for the client device 120 or (2) a translated set of access

21   control limits for that file 112. As described herein, the file server 110 validates trans-

22   lated user IDs for all Unix security style files 112, and preferably validates translated user

1   IDs for NT security style files 112 (when possible). Moreover, when the file server 110

2   validates translated access control limits for the file 112, the translated access control

3   limits are not recomputed for each request 121, but are cached with the file 112 for reuse.

4

5   The file server 110 thus also recognizes and enforces at least the following

6   circumstances:

7

8   o   NT security style. The file 112 has NT security style and has a corresponding set

9   of access control limits (an NT ACL), which has been set by a client device 120

10   using the CIFS file server protocol.

11

12   If a client device 120 makes a request 121 to access the file 112 using the

13   NFS file server protocol, the file server 110 determines the Unix user, associated with the

14   client device 120, that is making the request 121. The Unix user has a UID (user ID).

15

16   In a preferred embodiment, the file server 110 maps the Unix user into an

17   equivalent NT user. The file server 110 translates the UID into an SID (security ID) that

18   is an equivalent user to the Unix user. The file server 110 enforces the access control

19   limits (the NT ACL) for the equivalent NT user (the SID).

20

21   The file server 110 performs the following process to map the Unix user

22   into an equivalent NT user:

1

2   o   The client device 120 contacts the file server 110 using the NFS file server proto-

3   col. The NFS file server protocol request 121 includes a UID for the Unix user as-

4   sociated with the client device 120.

5

6   o   The file server 110 looks up the UID in the Unix password file (/etc/passwd), and

7   thus identifies the Unix user name for the Unix user. The Unix user name is an al-

8   phanumeric string identifying the Unix user.

9

10 o   The file server 110 translates the Unix user name into an NT user name using a

11   selected mapping file. Similar to the Unix user name, the NT user name is an al-

12   phanumeric string identifying the NT user. If there is no such translation for the

13   Unix user name, the file server 110 uses the Unix user name, without translation,

14   as the NT user name.

15

16   In a preferred embodiment, the mapping file includes a set of records each identi-

17   fying an NT user by NT user name, and associating an equivalent Unix user name

18   with the NT user name.

19

20 o   The file server 110 contacts an NT domain controller to determine an SID for the

21   NT user name. If there is no such NT user, the file server 110 uses a selected pa-

1   rameter for unmapped Unix users. In a preferred embodiment, this selected pa-

2   rameter is set to the NT user "guest" by default.

3

4   o   The file server 110 contacts the NT domain controller to obtain the SIDs of all

5   groups for which the NT user is a member.

6

7   The file server 110 caches UID-to-SID mappings for a period of time, pref-

8   erably about a few hours.

9

10   In an alternative preferred embodiment, or if the file server 110 is unable to

11 map Unix users to NT users (for example, if domain authentication has been turned off),

12 the file server 110 maps the NT ACL into a no less restrictive set of Unix Perms. The file

13 server 110 determines these Unix Perms in response to the NT ACL and in response to

14 the Unix user. The file server 110 enforces the mapped access control limits (the Unix

15 Perms) for the actual Unix user (the UID).

16

17   The file server 110 might perform dynamic permission mapping, in which

18 the file server 110 maps the NT ACL into a set of Unix Perms at the time the mapping is

19 required. In a preferred embodiment, the file server 110 caches the translated Unix

20 Perms with the file 112. Accordingly, for validating access control limits, the file server

21 110 performs static permission mapping, in which the file server 110 maps the NT ACL

22 into a set of Unix Perms at the time the NT ACL is set.

1

2   The file server 110 performs the following process to achieve static permis-

3   sion mapping:

4

5   o   The file server 110 determines the NT user that is the owner of the file 112, and

6   maps the NT user into an equivalent Unix user (the file server 110 maps the SID

7   for the NT user into a UID for a Unix user).

8

9   o   The file server 110 examines the NT ACL for the file 112 and determines whether

10   there are any "deny access" provisions.

11

12 o   If the NT ACL for the file 112 has no "deny access" provisions, the file server 110

13   generates a set of Unix Perms having an entry for "user permissions," consistent

14   with the file access control limits provided by the NT ACL. The file server 110

15   sets the Unix Perms for "group permissions" equal to the Unix Perms for "other

16   permissions." The file server 110 sets the Unix Perms for "other permissions"

17   equal to the NT ACL entry for "everyone," if one exists.

18

19 o   If the NT ACL for the file 112 does have "deny access" provisions, the file server

20   110 rejects the request 121.

21

1   Because static permission mapping is not responsive to the particular user

2   making the request 121, the file server 110 does not attempt to determine what the provi-

3   sions of the NT ACL are for that particular user.

4

5   o   <u>Unix security style.</u> The file 112 has Unix security style and has a corresponding

6   set of access control limits (Unix Perms), which have been set by a client device

7   120 using the NFS file server protocol

8

9   If a client device 120 makes a request 121 to access the file 112 using the

10   CIFS file server protocol, the file server 110 determines the NT user, associated with the

11   client device 120, that is making the request 121. The NT user has an SID (session ID).

12

13   The file server 110 maps the NT user into an equivalent Unix user. The file

14   server 110 translates the SID into a UID that is an equivalent user to the NT user. The

15   file server 110 enforces the access control limits (the Unix Perms) for the equivalent Unix

16   user (the UID).

17

18   The file server 110 performs the following process to map the NT user into

19   an equivalent Unix user:

20

1   o   The client device 120 starts a CIFS session (the client device 120 first contacts the

2   file server 110 using the CIFS file server protocol). The client device 120 trans-

3   mits its NT user name to the file server 110.

4

5   o   The file server 110 translates the NT user name into a Unix user name using a

6   mapping file. If there is no such translation for the NT user name, the file server

7   110 uses the NT user name, without translation, as the Unix user name.

8

9   o   The file server 110 looks up the Unix user name in the Unix password file

10   (/etc/passwd), and thus identifies the Unix user, the UID for the Unix user, the

11   Unix user's primary group, and the primary GID (group ID) for the Unix user. If

12   there is no such Unix user name in the Unix password file, the file server 110 uses

13   a selected parameter for unmapped NT users. In a preferred embodiment, this se-

14   lected parameter is set to the Unix user "nobody" by default.

15

16   o   The file server 110 looks up the Unix user name in the Unix group file

17   (/etc/groups) to determine any other groups and any other GIDs associated with

18   the Unix user.

19

20   *Reading and Modifying Access Control Limits*

21

1   Each file 112 has its security style set by the file server 110 so that either

2   (a) a request 121 to perform an operation on the file 112, or (b) a request 121 to perform

3   an operation that sets the access control limits for the file 112, produce expected results.

4

5   When the file 112 is first created, the file server 110 sets the security style

6   for the file 112 equal to a security style associated with the file server protocol used to

7   create it. (This is limited by restrictions imposed by access control trees, described

8   herein.) Thus, if the file 112 is created using the NFS file server protocol, the security

9   style for the file 112 is set to Unix security style. Similarly, if the file 112 is created us-

10   ing the CIFS file server protocol, the security style for the file 112 is set to NT security

11   style.

12

13   When the file 112 has its access control limits modified, the file server 110

14   sets the security style for the file 112 equal to a security style associated with the new ac-

15   cess control limits. (This is limited by restrictions imposed by access control trees, de-

16   scribed herein.) Thus, if a client device 120 sets a set of Unix Perms for the file 112, the

17   security style for the file 112 is set to Unix security style. Similarly, if a client device 120

18   sets an NT ACL for the file 112, the security style for the file 112 is set to NT security

19   style.

20

21   The file server 110 can receive a request 121 to read or view the access

22   control limits for a file 112. Also, when the file server 110 receives a request 121 to

22

1   make an incremental change to the access control limits for a file 112, it determines the

2   current access control limits for the file 112 before making the incremental change.

3

4   The file server 110 thus recognizes and enforces at least the following cir-

5   cumstances:

6

7   o   NT security style. The file 112 has NT security style and has a corresponding set

8   of access control limits (an NT ACL), which has been set by a client device 120

9   using the CIFS file server protocol.

10

11   If a client device 120 makes a request 121 to read or modify the access

12   control limits for the file 112 using the NFS file server protocol, the file server 110 de-

13   termines the Unix user, associated with the client device 120, that is making the request

14   121.

15

16   The file server 110 performs the same process to map an NT ACL into a set

17   of Unix Perms as described above for validation of file access control, with the following

18   exceptions:

19

20   Unlike validation of access control limits, the file server 110 treats transla-

21   tion of access control limits differently for requests 121 to read or modify the access

22   control limits for the file 112.

1

2   Preferably, the file server 110 performs dynamic permission mapping, in

3   which the file server 110 maps the NT ACL into a set of Unix Perms at the time the map-

4   ping is required. NT security style is richer than Unix security style—for example, Unix

5   security style has no "deny access" provisions. Thus, it is possible for the file server 110

6   to map the NT ACL into a set of Unix Perms that appears different for different Unix us-

7   ers. For example, if the NT ACL, for a file 112 whose owner is Charles, specifically

8   provides read access to Allen but specifically denies read access to Beth, the file server

9   110 will provide different Unix perms to each of the users Allen and Beth. One set will

10   allow read access by Allen's group and one set will disallow read access by Beth's group,

11   in harmony with the access provided by the actual NT ACL.

12

13   The file server 110 performs the following process to achieve dynamic

14   permission mapping:

15

16   o   The file server 110 determines the NT user that is the owner of the file 112, and

17   maps the NT user into an equivalent Unix user (the file server 110 maps the SID

18   for the NT user into a UID for a Unix user).

19

20   o   The file server 110 examines the NT ACL for the file 112 and determines whether

21   there are any "deny access" provisions.

22

1   o   If the NT ACL for the file 112 has no "deny access" provisions, the file server 110

2   generates a set of Unix Perms having entries for "user permissions" and "other

3   permissions," consistent with the file access control limits provided by the NT

4   ACL. The file server 110 sets the Unix Perms for "group permissions" equal to

5   the Unix Perms for "other permissions."

6

7   o   If the NT ACL for the file 112 does have "deny access" provisions, the file server

8   110 attempts to determine if any apply to the particular Unix user. If the file

9   server can tell, it generates a set of Unix Perms that reflect the access control lim-

10   its currently available for this particular file 112 and this particular Unix user. If

11   the file server 110 cannot tell, it rejects the request 121. (Alternatively, the file

12   server 110 could reject the request 121 if there are any "deny access" provisions in

13   the NT ACL.)

14

15   In alternative embodiments, the file server 110 may perform static permis-

16   sion checking, similar to validation of access control limits, for requests 121 to read or

17   modify the access control limits for the file 112.

18

19   If the request 121 attempts to modify attributes of the file 112 that have no

20   effect on access control limits for the file 112 (such as access time or modify time), the

21   file server 110 makes those modifications without change to the access control limits for

22   the file 112.

25

1

2   If the request 121 attempts to modify some but not all access control limits

3   for the file 112, the file server 110 generates a set of Unix Perms in response to the NT

4   ACL for the file 112, as described above. The file server 110 modifies the generated

5   Unix Perms as specified by the request 121. If the file server 110 cannot generate a set of

6   Unix Perms in response to the NT ACL for the file 112, the file server 110 rejects the re-

7   quest 121.

8

9   One difference in setting access control limits is that, according to NT secu-

10   rity style, files 112 can be specifically set to be "READ-ONLY." According to Unix se-

11   curity style, files are set to be read only by clearing the WRITE permission for the owner

12   of the file 112. When a client device 120 using the CIFS file server protocol attempts to

13   set the READ-ONLY attribute of a file 112 with Unix security style, the file server 110

14   clears the WRITE permission for the owner of the file 112 in the Unix Perms for that file

15   112.

16

17   o   Unix security style. The file 112 has Unix security style and has a corresponding

18   set of access control limits (Unix Perms), which have been set by a client device

19   120 using the NFS file server protocol

20

1   The file server 110 performs the following process to map a set of Unix

2   Perms into an NT ACL for display or modification of those Unix Perms by a CIFS client

3   device 120:

4

5   o   The file server 110 generates an NT ACL entry for "owner," providing the same

6   access control limits as the Unix Perms entry for "user permissions."

7

8   o   The file server 110 generates an NT ACL entry for "everyone," providing the

9   same access control limits as the Unix Perms entry for "other permissions."

10

11 o   If possible, the file server 110 generates an NT ACL entry for the actual request-

12   ing user, providing the same access control limits as the Unix Perms entry for that

13   user. This step could require mapping the Unix user into an equivalent NT user

14   using the UID-to-SID cache.

15

16   Similar to modification of an NT ACL entry by a Unix user, if the request

17 121 (for modification of Unix Perms by an NT user) attempts to modify attributes of the

18 file 112 that have no effect on access control limits for the file 112, the file server 110

19 makes those modifications without change to the access control limits for the file 112.

20

21   If the request 121 attempts to modify some but not all access control limits

22 for the file 112, the file server 110 generates an NT ACL in response to the set of Unix

27

1   Perms for the file 112, as described above. The file server 110 modifies the generated

2   NT ACL as specified by the request 121.

3

4   *Access Control Subtrees*

5

6   In a preferred embodiment, the files 112 in the file system 111 are organ-

7   ized into a tree, having a set of branch nodes and a set of leaf nodes. One branch node of

8   the tree is a root node, and each branch node of the tree is a root node for a subtree of the

9   tree. In the file system 111, each branch node is a directory, and each leaf node is a file

10   112. A directory is a type of file 112 that includes information about those branch nodes

11   and leaf nodes in a subtree for which it is the root node.

12

13   The file server 110 associates a limited set of access control models with

14   each subtree. In a preferred embodiment in which the file server 110 supports Unix secu-

15   rity style and NT security style, the file server 110 designates each subtree as being NT-

16   only format, Unix-only format, or mixed format.

17

18   <u>NT-only Format</u>

19

20   When the file server 110 designates a subtree as being NT-only format, it

21   restricts creation of files 112 within that subtree to files 112 having NT security style.

28

1   The file server 110 also prohibits changing the access control model of files 112 within

2   that subtree to other than NT security style.

3

4   According to NT security style, new files 112 inherit NT ACL settings

5   from their parent nodes. If a client device 120 using the NFS file server protocol at-

6   tempts to create a file 112 in a subtree having NT-only format, that file 112 can only be

7   created by the Unix user equivalent to the NT user who is the NT-owner of the root node

8   of the subtree. The file server 110 determines if the Unix user making the request 121 is

9   the equivalent by (a) mapping the SID for the NT user who is the owner into an equiva-

10   lent UID; (b) storing that UID in its record for the file 112; and (c) comparing that UID

11   with the UID in the request 121.

12

13   According to NT security style, there is a particular "DELETE" permission

14   and a particular "DELETE-CHILD" permission. If the file server 110 is unable to deter-

15   mine if a Unix user has these permissions, it rejects requests 121 to delete files 112 in

16   NT-only format subtrees, unless the request 121 is from the owner of the file 112 (the

17   equivalent Unix user of the NT user who is the owner) or the Unix user "root".

18

19   Acording to NT security style, there is a particular "CHANGE-

20   PERMISSION" permission and a particular "TAKE-OWNER" permission. If the file

21   server 110 is unable to determine if a Unix user has these permissions, it denies requests

22   121 to set any permissions for files 112 in a NT-only format subtree, unless the request

1 121 is from the owner of the file 112 (the equivalent Unix user of the NT user who is the

2 owner) or the Unix user "root".

3

4 <u>Unix-only Format</u>

5

6   Similarly, when the file server 110 designates a subtree as being Unix-only

7 format, it restricts creation of files 111 within that subtree to files 111 having Unix secu-

8 rity style. The file server 110 also prohibits changing the access control model of files

9 111 within that subtree to other than Unix security style. Attempts to set an NT ACL

10 would change the access control model for that file 112 to NT security style, and so are

11 rejected in a Unix-only format subtree.

12

13   When a client device 120 using the CIFS file server protocol creates a file

14 112 in a Unix-only format subtree, the file server 110 sets the owner of the file 112 to the

15 Unix user equivalent to the NT user making the request 121. The file server 110 maps

16 the SID for the NT user to a UID for an equivalent Unix user, and uses that UID to set the

17 owner of the file 112.

18

19   According to Unix security style, there is no "CHANGE-PERMISSION"

20 permission or "TAKE-OWNER" permission. The file server 110 always denies requests

21 121 to set these permissions for files 112 in a Unix-only format subtree.

22

1   Mixed Format

2

3   When the file server 110 designates a subtree as being mixed format, it al-

4. lows creation of files 111 with either Unix security style or NT security style. The file

5   server 110 does not prohibit changing the access control model of files 111 within that

6   subtree to either Unix security style or NT security style.

7

8   An administrator for the file server 110 can change the designation of a

9   subtree from a first format to a second format (for example, from mixed format to either

10   NT-only format or Unix-only format). When the second format is possibly incompatible

11   with the first format (for example, a subtree changed to NT-only format includes nodes

12   that are Unix security style), the file server 110 converts those files 112 with incompati-

13   ble access control models as it sets permissions for those files 112. Requests 121 for a

14   file 112 which only check permissions are still validated using the access control model

15   in place for the file 112.

16

17   Although the invention is described herein with regard to only two access

18   control models, the invention can readily be used with three or more access control mod-

19   els. In such alternative embodiments, there are a larger number of possible subtree for-

20   mats, including subtree formats that restrict the files 112 within that subtree to one of a

21   set of multiple access control models, less than the set of all access control models recog-

22   nized by the file server 110.

1

2   In a preferred embodiment, the root node of the file system 111 is desig-

3   nated as having mixed format. Client devices 120 that are owners of a subtree can mod-

4   ify the format of a subtree by request 121 to the file server 110; thus, client devices 120

5   can modify subtrees to have NT-only format, Unix-only format, mixed format. When a

6   new subtree is created, the file server 110 designates the new subtree as having the same

7   format as its parent; thus, mixed format if it is created within a subtree that is already

8   mixed format (the default), NT-only format if it is created within a subtree that is already

9   NT-only format, and Unix-only format if it is created within a subtree that is already

10   Unix-only format.

11

12   *Alternative Embodiments*

13

14   Although preferred embodiments are disclosed herein, many variations are

15   possible which remain within the concept, scope, and spirit of the invention, and these

16   variations would become clear to those skilled in the art after perusal of this application.